Chapter 1：OpenCV X 人臉辨識

本章重點

使用 OpenCV 來辨識人臉與訓練特定的人臉。

準備材料

可上網的電腦（含攝影鏡頭）

學習目標

1. 安裝虛擬環境。
2. 安裝 OpenCV。
3. OpenCV 的基本使用。
4. OpenCV 的人臉辨識。
5. 整合專案 1：多個特定人臉辨識。
6. 整合專案 2：物聯網應用，將目標物影像即時回傳 – 以 LINE 為例。

1-1. 安裝虛擬環境

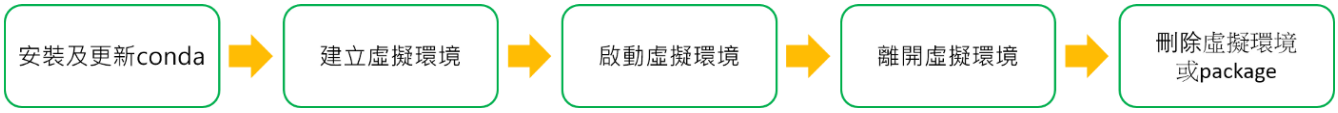
利用 conda 為每一個不同需求的專案，建立一個獨立適合的虛擬環境。

用 conda 建立及管理 Python 虛擬環境

開發 Python 的專案時，最常遇見的問題就是不同專案可能會有不同的 Python 版本，以及不同的 Package (套件) 需要安裝，例如：Python 2 或 Python 3。那麼管理不同版本的專案會是一個問題。如果你只需要使用特定的套件，或是想要嘗試各種不同的環境應用，但又不想彼此的開發環境受到影響，那 Anaconda 的套件管理系統 conda 將會是一個不錯的解決方案。

如果習慣用 Python 的軟體包管理系統 pip 的人，對於 conda 的指令一定也可以馬上上手，因為它和 pip 指令非常的相似，conda 是套件管理系統，也可用來建立虛擬環境。

conda 命令是管理在安裝不同 package 時的主要介面，使用 conda 時，你可以進行建立 (create)、輸出 (export)、列表 (list)、移除 (remove) 和更新 (update) 環境於不同 Python 版本及 Packages，同時也可以分享你的虛擬環境。接下來將使用 conda 建立及管理虛擬環境，有下列五個步驟：



Step 1 安裝及更新 conda。

Anaconda 下載：https://www.anaconda.com/products/individual

安裝好 Anaconda 後，在 Windows 開始選單 (start menu) 中選擇『 Anaconda Prompt 』，並輸入以下常用的指令。

檢查目前 Python 版本：

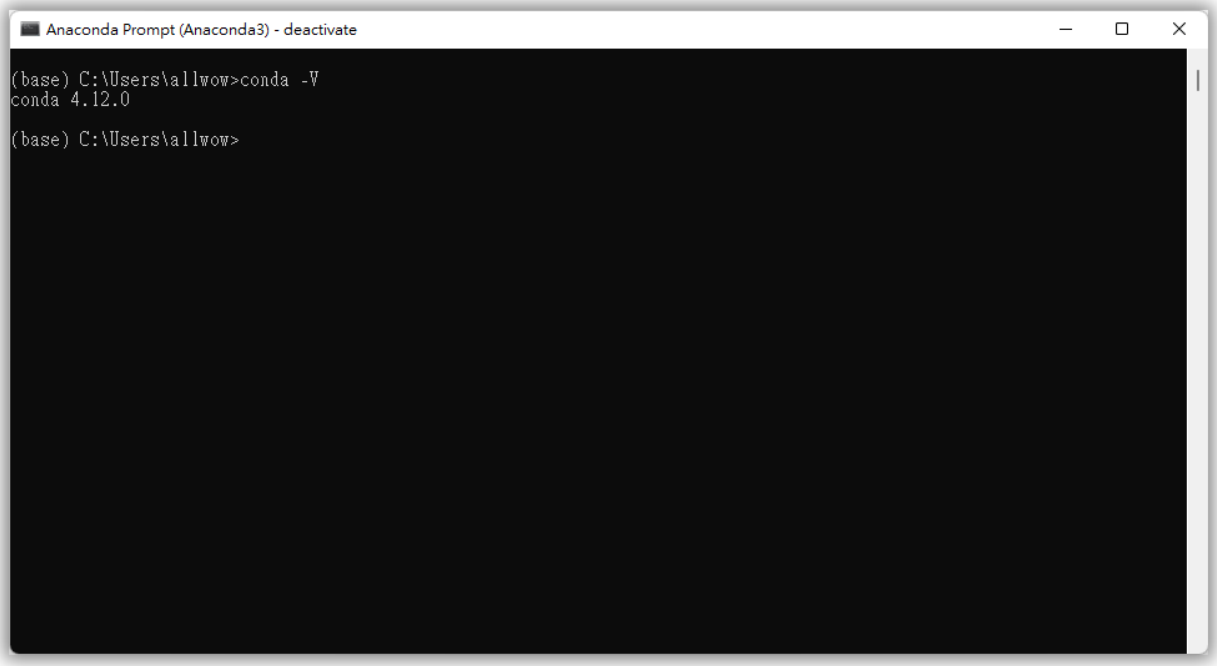
```
python -V
```

檢查目前 Python 環境安裝了那些 Package：

```
pip list
```

檢查目前 conda 版本：

```
conda –V
```



檢查目前 conda 環境安裝了那些 Package：(也會列出 Python 目前安裝的 Package)

```
conda list
```

若想要進行更新，可以輸入下列命令：(更新時必須以系統管理員執行 Anaconda Prompt)

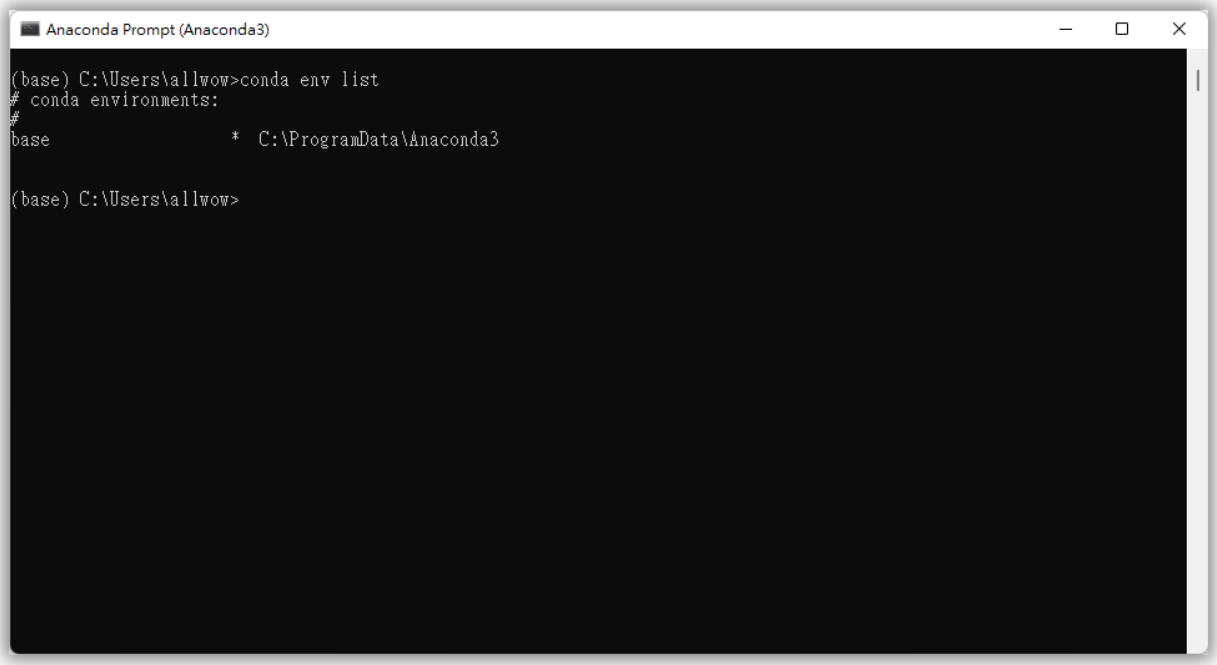
```
conda update conda
```

Step 2 建立虛擬環境。

看目前系統已經安裝幾個虛擬環境：

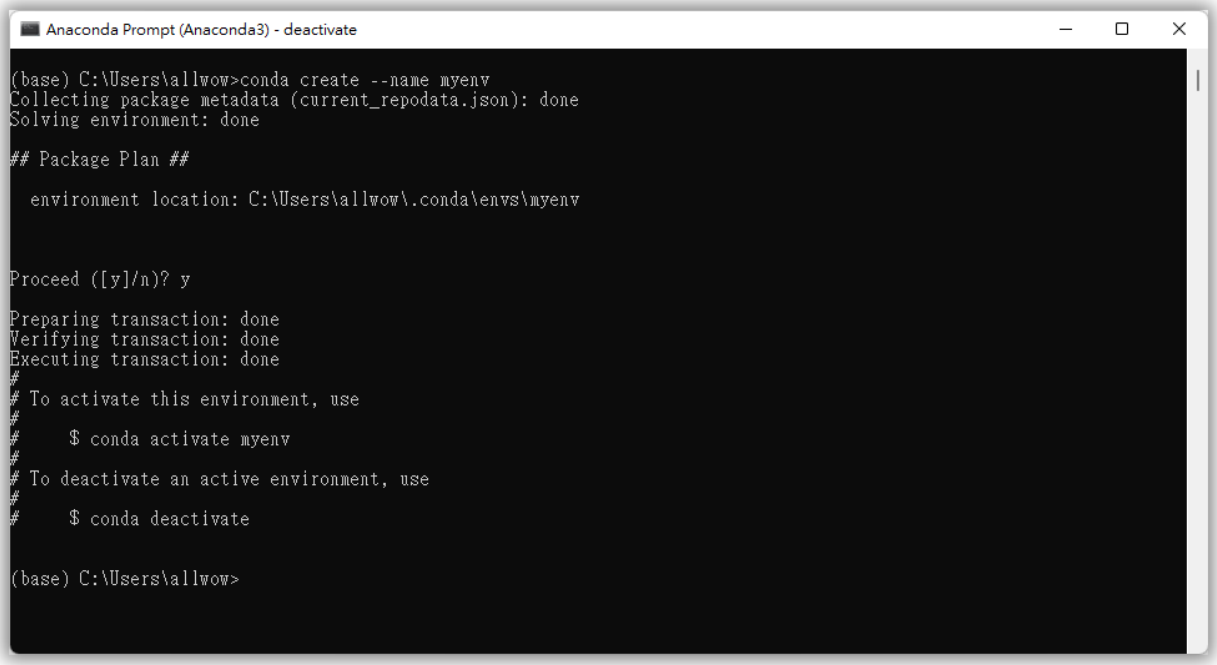
```
conda env list
```

預設只有一個環境，為系統預設的 base：



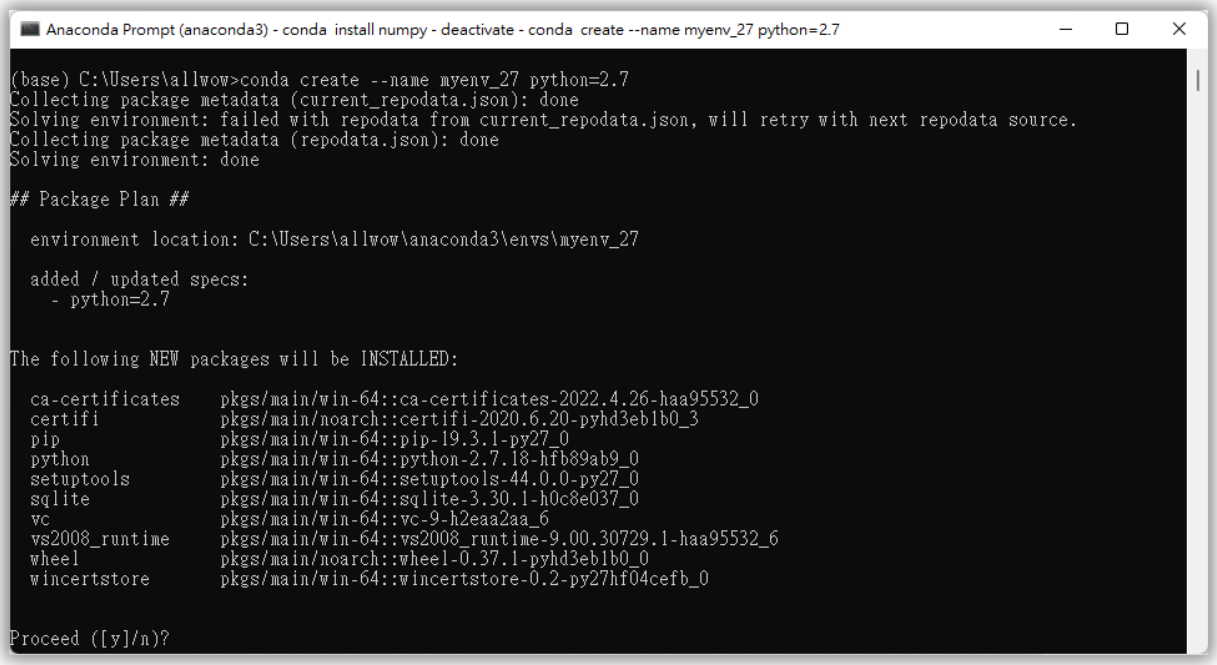
若要建立一個叫做 myenv_39 的虛擬環境，並且指定 Python 3.9 的版本：

```
conda create --name myenv_39 python=3.9
```

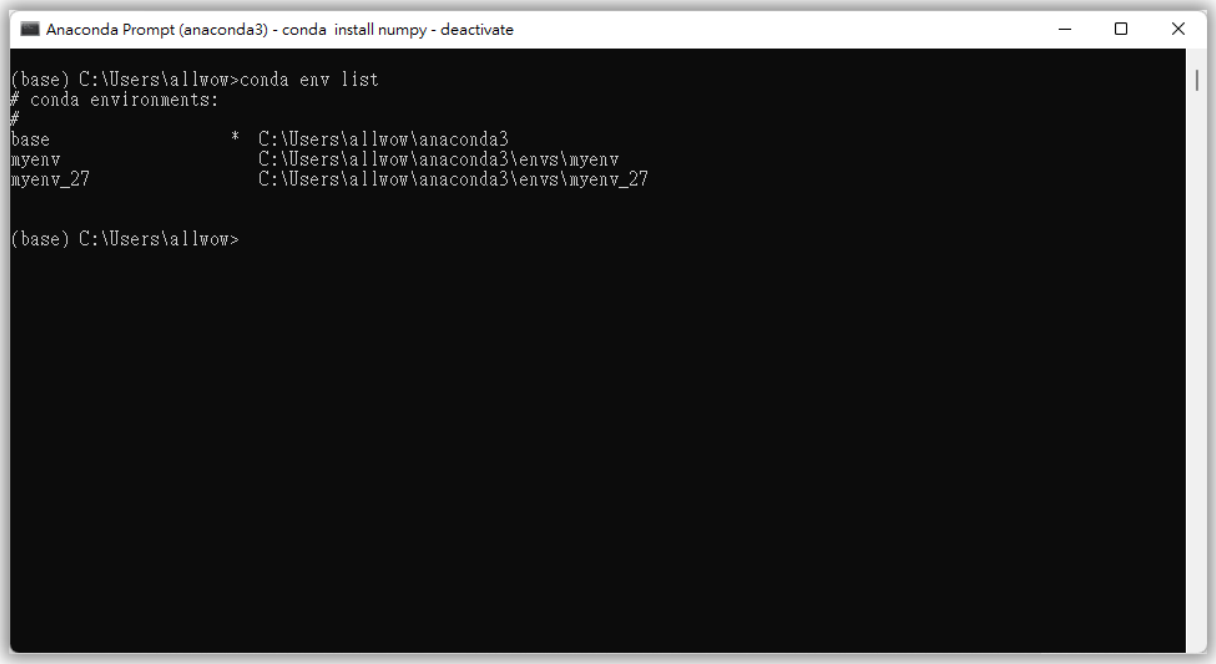


若要建立一個叫做 myenv_27 的虛擬環境，並且指定 Python 2.7 的版本：

```
conda create --name myenv_27 python=2.7
```



再次執行 conda env list，列出目前虛擬環境狀況，將會看到多了兩個剛建立的虛擬環境 myenv 與 myenv_27。



Step 3 啟動虛擬環境。

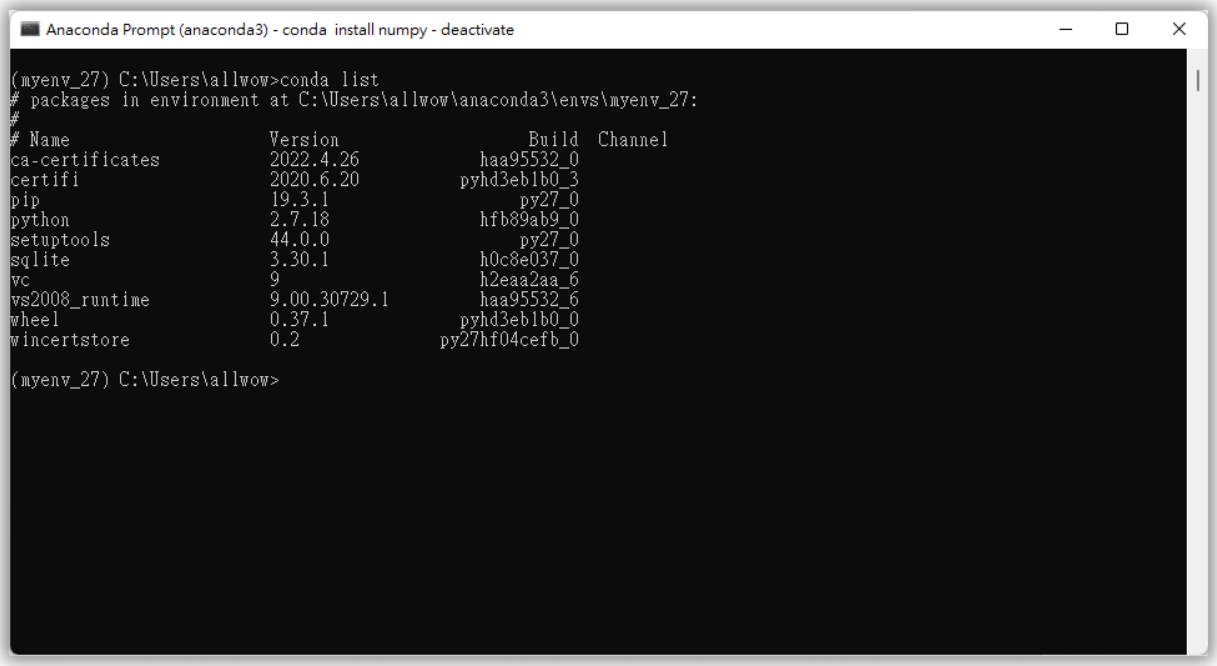
啟動一個新的虛擬環境：

```
activate myenv_27
```

這時候 cmd 模式下前面會有一個 (myenv_27)，表示你目前是處於此虛擬環境中，這時候你就可以在此虛擬環境中，開始安裝你所需要的各種 Package。

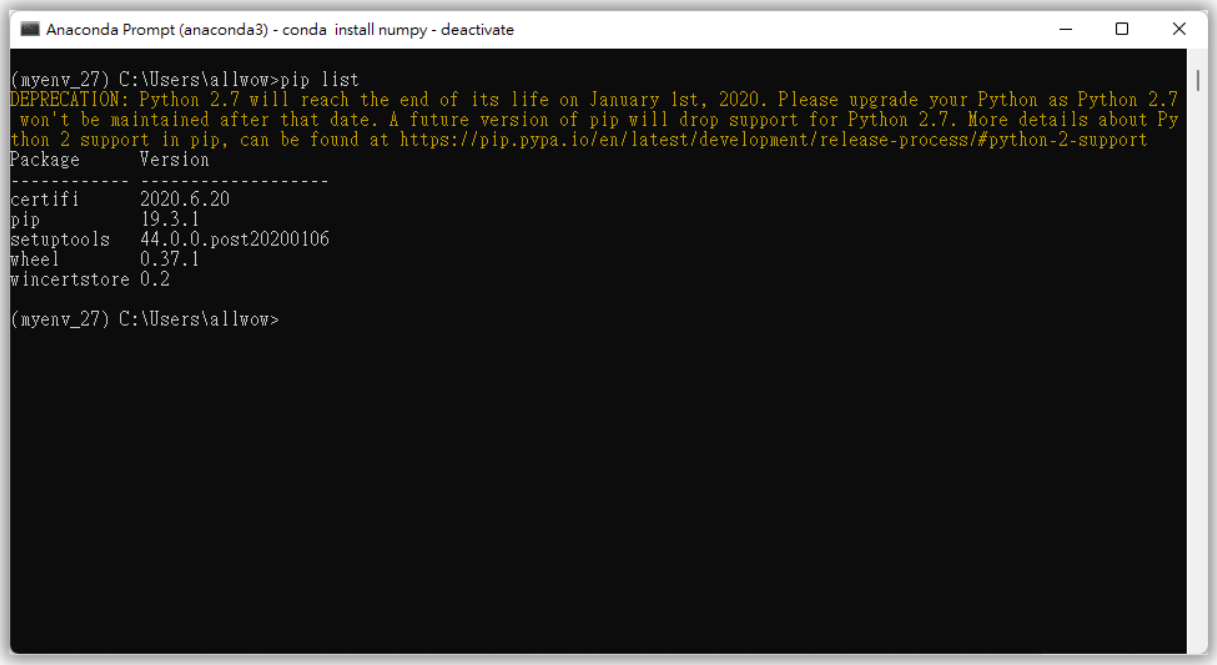
檢查目前此虛擬環境中的 conda 安裝了那些東西：

```
(myenv_27) conda list
```



檢查目前虛擬環境的 Python 環境，安裝了那些 Package：

```
(myenv_27) pip list
```



要在此虛擬環境下安裝所需套件，例如：numpy。

```
(myenv_27) conda install numpy
```

Step 4 離開虛擬環境。

關閉目前的虛擬環境：

```
(myenv_27) deactivate
```

Step 5 刪除虛擬環境的 Package 或虛擬環境本身。

要刪除虛擬環境中某個 Package：(例如：虛擬環境 myenv_27 中的 numpy)

```
(myenv_27) conda remove --name myenv_27 numpy
```

要刪除整個虛擬環境：(不能刪除當前的虛擬環境，必須先關閉目前的虛擬環境)

```
conda env remove --name myenv_27
```

Step 6 複製虛擬環境。

複製虛擬環境：(myEnvNameDes：目的檔案，myEnvNameSou：來源檔案)

```
conda create -n myEnvNameDes --clone myEnvNameSou
```

例如：從已存在的 myenv 複製出一個名為 myenv_new 的虛擬環境。

```
conda create -n myenv_new --clone myenv
```

Step 7 為虛擬環境安裝 Anaconda。

Anaconda 的 Spyder 有專業的除錯模式，可單步執行追蹤程式碼。目前 Anaconda 的最新版本是支援 Python 3.9，所以先創建一個 Python 3.9 的虛擬環境，然後安裝 Anaconda 整合開發環境後，再安裝各種 Python 的 Package，這樣的虛擬環境會比較方便專案管理。

建一個 Python 3.9 的虛擬環境 myenv_39，並且切換至 myenv_39：

```
conda create --name myenv_39 python=3.9 & activate myenv_39
```

安裝 Anaconda：

```
(myenv_39) conda install anaconda
```

建立好基本的虛擬開發環境後，以後就可以使用複製虛擬環境的方式，快速管理。

1-2. 安裝 OpenCV

OpenCV 介紹

OpenCV (Open Source Computer Vision Library) 是跨平台 BSD 授權的一套著名的電腦視覺函式庫，這是 Intel 發起並參與開發，幫助開發圖片處理、影片處理、電腦視覺的人臉辨識和物體辨識等人工智慧的相關應用。

- OpenCV 也可以在 Android、iOS 上開發，支援的程式語言也有 C/C++、Java、Python，重點是免費的。

基本上，安裝 OpenCV 有兩種方式，如下所示：

- 使用 pip 指令：使用 Python 套件管理 pip 安裝已編譯好的 OpenCV，安裝過程比較簡單，但是不會優化 OpenCV 的執行效能。(本次採此種安裝方式)
- 自行編譯 OpenCV 的原始碼進行安裝：如果有特殊 OpenCV 版本和優化需求，則可自行編譯 OpenCV 的原始碼。

以 pip 安裝 OpenCV 套件

一般來說，較新的 OpenCV 版本需要較新的 NumPy 版本來保持兼容性，推薦版本匹配：

OpenCV 版本	推薦的 numpy 版本
4.5.x - 4.8.x	1.21.x - 1.25.x
4.4.x 及更早版本	1.19.x - 1.20.x
3.x 系列	1.16.x - 1.18.x

首先，建立一個 OpenCV 專用的 Python 3.9 虛擬環境，可由上一節完成的 myenv_39 來複製，名稱為 opencvenv_39：

```
conda create -n opencvenv_39 --clone myenv_39
```

載入 Python 虛擬環境 opencvenv：

```
activate opencvenv_39
```

接下來，在虛擬環境 opencvenv 中進行以下三個安裝步驟：

Step 1 安裝 OpenCV 的相關支援模組。

以 python 3.9 為例：

安裝支援的 Python 模組，numpy 為陣列容器 (一種存放物件的資料結構)、matplotlib 為計算 numpy 的繪圖庫、imutils 可以讓我們更容易使用 OpenCV 圖片處理：

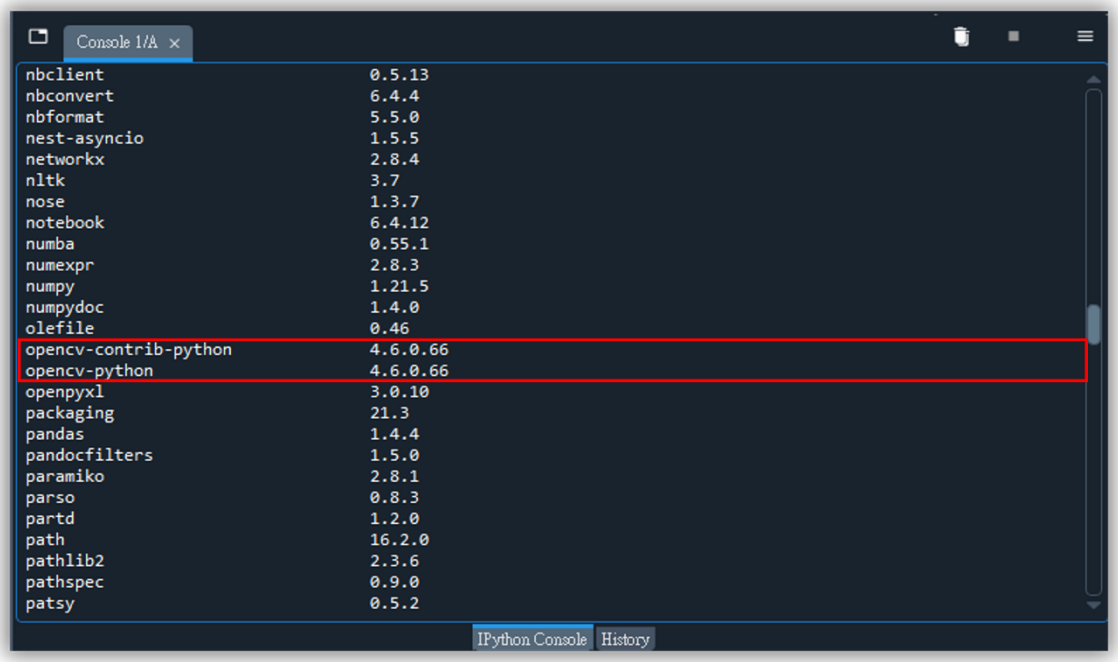
```
(opencvenv_39) pip install numpy==1.21.4    # 陣列容器，用於影像數值計算
(opencvenv_39) pip install matplotlib    # 畫圖用
(opencvenv_39) pip install imutils    # 常用的圖片處理模組
```

Step 2 安裝 OpenCV。

```
(opencvenv_39) pip install opencv-python    # OpenCV 模組
(opencvenv_39) pip install opencv-contrib-python    # OpenCV 擴充模組
```

輸入指令來檢視虛擬環境安裝的套件清單：

```
(opencvenv_39) pip list
```



Step 3 檢查 Python 是否安裝好 OpenCV。

在 Spyder 的 『 IPython Console 』中匯入 OpenCV 的模組名稱 cv2，並查詢版本：

```
>>> import cv2    # 匯入 OpencCV 模組
>>> cv2.__version__    # 使用 .__version__ 屬性顯示 OpenCV 版本
Out[2]: '4.6.0'    # 目前的版本為 4.6.0
```

1-3. OpenCV 的基本使用

圖片處理

讀取與顯示圖片

Python 程式是呼叫 OpenCV 的 `imread()` 方法讀取圖片，和 `imshow()` 方法顯示圖片。

1-1 讀取與顯示圖片。

```
import cv2

img = cv2.imread('images/koala.jpg')
cv2.imshow('Koala', img)

gray_img = cv2.imread('images/koala.jpg', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Koala:gray', gray_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

`imread()` 有屬性可設定以下的圖片色彩：

cv2.IMREAD_COLOR	彩色，預設值。
cv2.IMREAD_UNCHANGED	沒有改變，包含透明度的圖片內容。
cv2.IMREAD_GRAYSCALE	灰階。

程式設計師的思考

物件.屬性：不是方法，當宣告一個物件實體後，會有相關的物件靜態或特徵描述，稱為屬性。這樣可以簡化物件的使用，物件的詳細說明請參考『 物件導向程式的類別設計 』。

取得圖片資訊

我們可以使用 shape 屬性取得圖片資訊的尺寸和色彩數，例如：分別讀取成彩色和灰階圖片後，顯示圖片資訊。

1-1a 取得圖片資訊。

```
import cv2

img = cv2.imread('images/koala.jpg')
img2 = cv2.imread('images/koala.jpg', cv2.IMREAD_GRAYSCALE)
print(img.shape)
print(img2.shape)

h, w, c = img.shape
print('圖片高: ', h)
print('圖片寬: ', w)
```

調整圖片尺寸

OpenCV 的 cv2.resize() 在調整圖片尺寸時，會改變圖片的長寬比例，改用 imutils 模組的方法來自動調整圖片尺寸。

- 呼叫 imutils.resize() 方法調整圖片尺寸。

1-1b 調整圖片尺寸。

```
import cv2, imutils

img = cv2.imread('images/koala.jpg')
print(img.shape)
resized_img = imutils.resize(img, width = 300)
print(resized_img.shape)

cv2.imshow('Koala', img)
cv2.imshow('Koala:resized', resized_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Python 的資料結構

內建的資料結構

資料結構又稱為容器，用來存放多種物件，容器也是一種物件。

1. 字串 (string)：由字元組成。例如：myWord = '52python'。
   - myWord[0]：字串以索引取值，結果為：'5'。
2. tuple：由資料物件組成。例如： myTuple = (1, '2', 3)。
   - myTuple[0]：tuple 以索引取值，結果為：1。
3. 串列 (list)：由資料物件組成。例如：myList = [1, '2', 3]。
   - myList[1]：串列以索引取值，結果為：'2'。
4. 集合 (set)：由資料物件組成。例如：mySet = {1, '2', 3}。
   - 集合為無序所以沒有索引取值。
5. 字典 (dict)：由資料物件組成，以鍵：值表示。例如：myDick = {'A':1, 'B':'2', 'C':3}。
   - myDick['B']：字典以鍵取值，結果為：'2'。

第三方的資料結構

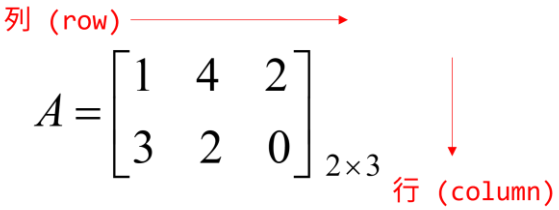
Numpy：Python 的擴充模組，常用於機器學習的資料處理。

```
>>> import numpy as np    # 匯入 numpy，並以 np 為簡寫代表
>>> a = np.array([10, 2, 45, 32, 24])    # 建立五個元素 1-D 陣列
>>> len(a)    # 計算元素個數
>>> a[2:4]    # 取出第 3 (a[2])、4 (a[3]) 的元素 (口訣：有頭無尾)
>>> a[:4]    # 取出第 1 (a[0]) ~ 3 (a[2]) 的元素
```

矩陣的切片

一維的稱為陣列 (array)，二維以上的稱為矩陣 (matrix)。以下介紹二維矩陣的切片：

```
import numpy as np
A = np.array([[1, 4, 2], [3, 2, 0]])
B = A[1:, 1:3]    # 二維矩陣的切片，第一個索引以列切片，第二個索引以行為切片
```



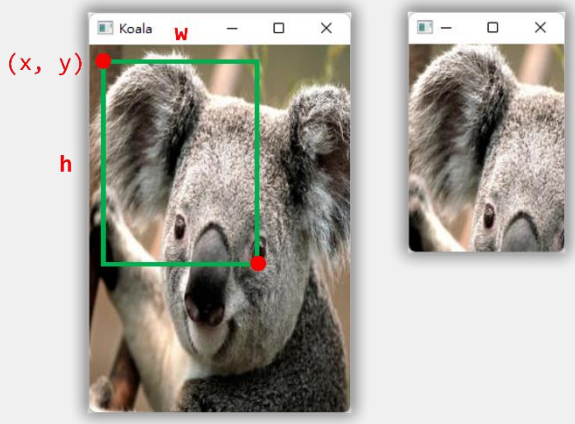
$$B = \begin{bmatrix} 2 & 0 \end{bmatrix}_{1\times2}$$

剪裁圖片

在 OpenCV 使用 imread() 方法讀取的圖片內容是一個 Numpy 矩陣，剪裁圖片就是在切割矩陣。

1-1c 剪裁圖片。

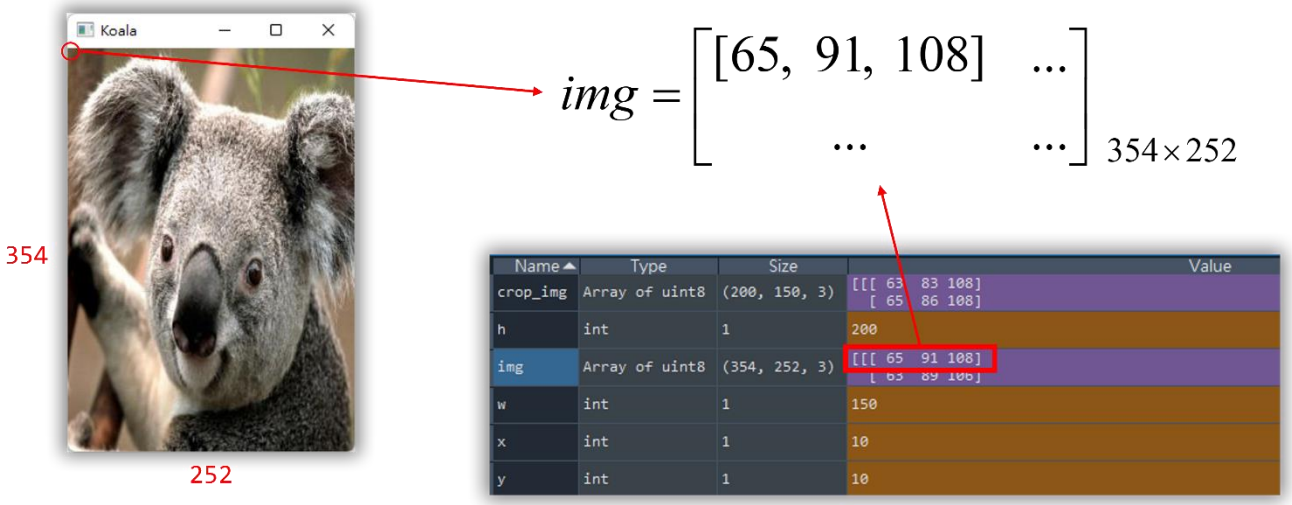
```
import cv2

img = cv2.imread('images/koala.jpg')
print(img.shape)

x = 10; y = 10
w = 150; h= 200

crop_img = img[y:y + h, x:x + w]
cv2.imshow('Koala', img)
cv2.imshow('corp_Koala', crop_img)
print(crop_img.shape)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



深入了解儲存全彩影像的矩陣

電腦以 RGB (紅綠藍) 三原色作為色彩顯示，每個原色可以設定 0 ~ 255。

- (R, G, B) = (0, 0, 0) 代表黑色；(R, G, B) = (255, 255, 255) 代表白色。
- 每一個像素可以用 RGB 表示來 256 x 256 x 256 = 16,777,216，這麼多的色彩數。



旋轉、翻轉和位移圖片

OpenCV 沒有旋轉和位移圖片的方法 (只有 cv2.flip() 方法來翻轉圖片)，需自行運算來旋轉和位移圖片，imutils 模組有提供相關方法來旋轉、位移圖片。

1-1d 旋轉、翻轉和位移圖片。

```
import cv2, imutils

img = cv2.imread('images/koala.jpg')
rotated_img = imutils.rotate(img, angle = 90)
fliped_img = cv2.flip(img, -1)
translated_img = imutils.translate(img, 25, -75)

cv2.imshow('Koala', img)
cv2.imshow('Koala:rotated', rotated_img)
cv2.imshow('Koala:fliped', fliped_img)
cv2.imshow('Koala:translated', translated_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

轉換成灰階和 BGR 圖片

OpenCV 讀取圖片後，可以呼叫 cvtColor() 方法轉換彩色圖片成為灰階圖片。

- 第 1 個參數：來源圖片，第 2 個參數：轉換色彩。
- OpenCV 預設的圖片色彩是 BGR，使用 cv2.COLOR_RGB2BGR 將 BGR 轉換成灰階。
- 若需要 RGB 格式，可以使用參數 cv2.COLOR_BGR2RGB 將 BGR 轉換成 RGB。

1-1e 轉換成灰階和 BGR 圖片。

```
import cv2

img = cv2.imread('images/koala.jpg')
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
bgr_img = cv2.cvtColor(rgb_img, cv2.COLOR_RGB2BGR)

cv2.imshow('Koala:original', img)
cv2.imshow('Koala:gray', gray_img)
cv2.imshow('Koala:rgb', rgb_img)
cv2.imshow('Koala:bgr', bgr_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

- BGR 和 RGB 不是顏色空間，它們只是不同的顏色排列順序。
- cv2.imread() 與 cv2.imshow() 預設處理的都是 BGR，所以不需要特別的轉換色彩。

從 URL 取得圖片

在 imutils 提供 url_to_image() 方法，可以讓我們直接從網路讀取圖檔內容，如下所示：

1-1f 從 URL 取得圖片。

```
import cv2, imutils

url = 'https://fchart.github.io/img/koala.png'
img = imutils.url_to_image(url)
cv2.imshow('Koala', img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

註記圖片

註記圖片就是在圖片上繪圖，我們可以在圖片上畫線、畫長方形、畫圓形、畫橢圓形和加上文字內容，如下所示：

1-1g 註記圖片。

```
import cv2

img = cv2.imread('images/koala.jpg')
cv2.line(img, (0, 0), (200, 200), (0, 0, 255), 5)
cv2.rectangle(img, (20, 70), (120, 160), (0, 255, 0), 2)
cv2.rectangle(img, (40, 80), (100, 140), (255, 0, 0), -1)
cv2.circle(img, (90, 210), 30, (0, 255, 255), 3)
cv2.circle(img, (140, 170), 15, (255, 0, 0), -1)
cv2.putText(img, 'OpenCV', (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 1,
            (0, 255, 255), 5, cv2.LINE_AA)

cv2.imshow('Koala', img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

寫入圖片

OpenCV 可以呼叫 imwrite() 方法將圖片內容寫入圖檔，如下所示：

1-1h 寫入圖片。

```
import cv2

img = cv2.imread('images/koala.jpg')
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite('images/result_gray.jpg', gray_img)
cv2.imwrite('images/result.png', img)
```

影像處理

影片就是一種動態影像，由連續的靜態影像圖片所組成，每一個靜態影像稱為『 影格 』(Frame，或稱**幀**)，每秒播放的靜態影像圖片數稱為『 影格率 』 (Frame per Second，或稱**幀率**)。

- OpenCV 支援讀取和播放影片檔案。

播放影片檔

Python 程式是建立 OpenCV 的 VideoCapture 物件來播放影片檔：

1-2 播放影片檔。

```
import cv2

cap = cv2.VideoCapture('images/YouTube.mp4')

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print('影片播放完畢')
        break
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) == 27:    # Windows 的『 ESC 鍵 』的鍵碼為 27
        break

cap.release()
cv2.destroyAllWindows()
```

影片處理顯示灰階影片

1-2a 將全彩影片以灰階播放。

```
import cv2

cap = cv2.VideoCapture('images/YouTube.mp4')

while cap.isOpened():
    ret, frame = cap.read()

    if not ret:
        print('影片播放完畢')
        break
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        cv2.imshow('frame', gray_frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

取得網路攝影機的影像

OpenCV 的 VideoCapture 物件除了開啟影片檔案，也可以開啟攝影機：

1-3 取得網路攝影機的影像。

```
import cv2

cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

更改影像的解析度

在 Python 程式建立 VideoCapture 物件後，可以呼叫 set 方法更改影片的寬、高和影格率：

1-3a 更改影像的解析度。

```
import cv2

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 180)
cap.set(cv2.CAP_PROP_FPS, 25)

while cap.isOpened():
    ret, frame = cap.read()
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

將影像寫入影片檔案

OpenCV 可以建立 VideoWrite 物件來寫入影片檔案：

1-3b 將影像寫入影片檔案。

```
import cv2

cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('images/output.avi', fourcc, 20, (640, 480))

while cap.isOpened():
    ret, frame = cap.read()

    if ret == True:
        out.write(frame)
        cv2.imshow('frame', frame)

        if cv2.waitKey(1) == 27:
            break
    else:
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

以下是常用編碼的設定參數：

編碼名稱	編碼字串	影片副檔名
YUV	*'I420'	.avi
MPEG-I	*'PIMT'	.avi
MPEG-4	*'XVID'	.avi
MP4	*'MP4V'	.mp4
Ogg Vorbis	*'THEO'	.ogv

1-4. OpenCV 的人臉辨識

OpenCV 內建的是哈爾特徵 (Haar-like feature) 演算法。

- 速度快，非類神經網路，適合低階的樹莓派。
- 使用哈爾特徵前，先下載人臉偵測的聯集分類器。

分類器下載：

- https://github.com/opencv/opencv/tree/master/data/haarcascades

以下是跟人臉有關的：

分類器	說明
haarcascade_frontalface_default.xml	人臉正面與側面
haarcascade_frontalface_alt2.xml	人臉正面效果比較好
haarcascade_profileface.xml	人臉側面效果比較好
haarcascade_eye.xml	偵測眼睛

使用內建的人臉辨識分類器

圖片的人臉辨識

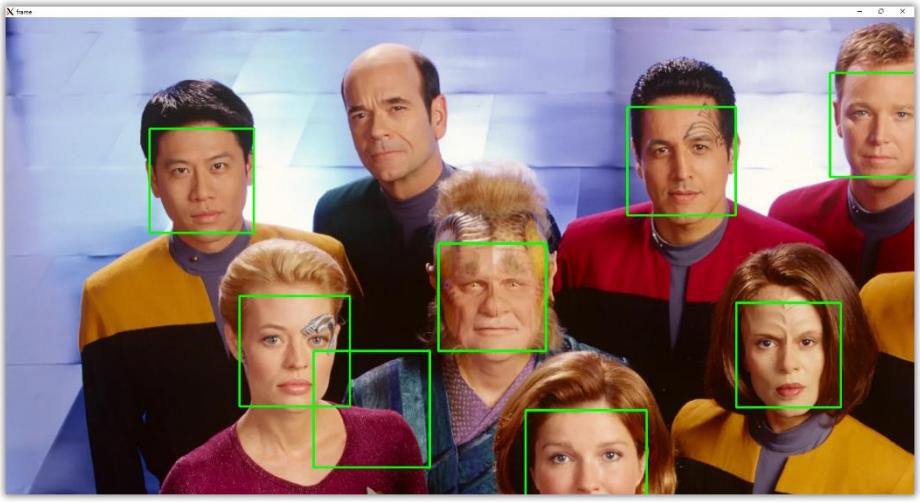
先針對靜態圖片進行人臉辨識，並調整參數：

1-4 單張圖片的人臉辨識。

```
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
image = cv2.imread('images/demo.jpeg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.1, 3)
# faces = face_cascade.detectMultiScale(gray, 1.05, 4)

for (x, y, w, h) in faces:
    image = cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 3)

cv2.namedWindow('preview', cv2.WINDOW_NORMAL)
cv2.imshow('preview', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

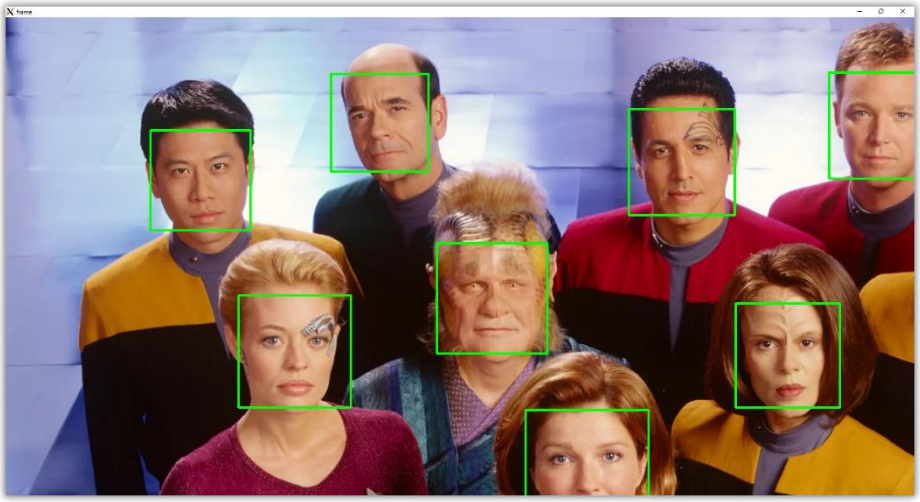
1-4 的辨識結果，主要是由以下的設定來調整的：

```
faces = face_cascade.detectMultiScale(gray, 1.1, 3)
```



當有人臉沒被辨識，需要調整參數：

```
faces = face_cascade.detectMultiScale(gray, 1.05, 4)
```



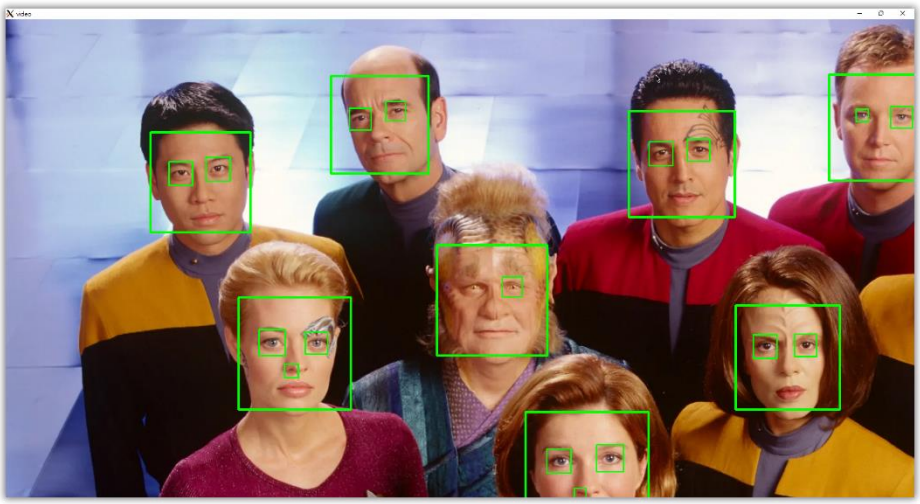
圖片的眼睛辨識

辨識完人臉後，可接著辨識眼睛的部分：

1-5 單張圖片的人臉與眼睛辨識。

```
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt2.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
image = cv2.imread('images/demo.jpeg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# faces = face_cascade.detectMultiScale(gray, 1.1, 3)
faces = face_cascade.detectMultiScale(gray, 1.05, 4)

for (x, y, w, h) in faces:
    image = cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 3)
    face_rect = gray[y:y + h, x:x + w]
    eyes = eye_cascade.detectMultiScale(face_rect, 1.15, 8)
    for (ex, ey, ew, eh) in eyes:
        image = cv2.rectangle(imges, (x + ex, y + ey), (x + ex + ew,
                                y + ey + eh), (0, 255, 0), 2)

cv2.namedWindow('preview', cv2.WINDOW_NORMAL)
cv2.imshow('preview', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

輸出結果：



即時影像的人臉辨識

只要將 `VideoCapture(0)` 加入，就可以使用攝影機進行即時人臉辨識。0 代表第一台攝影機，1 代表第二台攝影機，依此類推：

1-6 即時影像的人臉辨識。

```
import cv2

faceCascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.1,
                                minNeighbors = 5, minSize = (30, 30))
    print('人臉數: ', len(faces))

    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x, y), (x + w, y + h),
                                (0, 255, 0), 2)

    cv2.imshow('preview', frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

即時影像的人臉與眼睛辨識

再加入眼睛的辨識：

1-7 即時的人臉與眼睛辨識。

```
import cv2

faceCascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.1,
                                minNeighbors = 5, minSize = (30, 30))
    print('人臉數: ', len(faces))
    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x, y), (x + w, y + h),
                                (0, 255, 0), 2)
        face_rect = gray[y:y + h, x:x + w]
        eyes = eye_cascade.detectMultiScale(face_rect, 1.3, 8)
        for (ex, ey, ew, eh) in eyes:
            frame = cv2.rectangle(frame, (x + ex, y + ey), (x + ex + ew,
                                    y + ey + eh), (0, 255, 0), 2)

    cv2.imshow('preview', frame)

    if cv2.waitKey(1) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

特定人臉辨識

藉由專門分析人臉特徵的演算法找出的特徵值，然後跟已經儲存的特徵值比對，判定是誰的臉。

分為三階段：取樣、訓練、辨識。

- 取樣：收集訓練用的人臉圖片，建議每人有 100 張才有比較好的準確率。
- 訓練：OpenCV 提供三種演算法，Eigen、Fisher、LBPH。
- 辨識：依照訓練完的資料，判斷目前攝影機看到的人臉屬於哪一位。

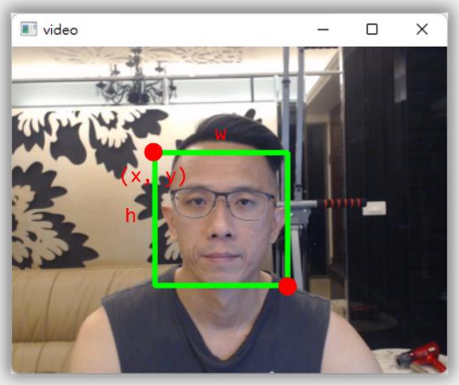
Step 1 取樣階段。

透過程式碼來自動抓取至少 100 張人臉圖片，當作訓練的樣本，這樣比用相機拍攝 100 張圖片要來的方便快速：

1-8-1 人臉取樣。(1-8-1-capture.py)

```
import cv2

ESC = 27
n = 1
index = 0
total = 100

def saveImage(face_image, index):
    filename = 'images/h0/{:03d}.pgm'.format(index)
    cv2.imwrite(filename, face_image)
    print(filename)

face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
                                cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)
cv2.namedWindow('video', cv2.WINDOW_NORMAL)

while n > 0:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    faces = face_cascade.detectMultiScale(gray, 1.1, 3)

    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x,y), (x + w,y + h), (0,255,0), 2)
        if n % 5 == 0:
            face_img = gray[y: y + h, x: x + w]
            face_img = cv2.resize(face_img, (400, 400))
            saveImage(face_img, index)
            index += 1

            if index >= total:
                print('get training data done')
                n = -1
                break

        n += 1

    cv2.imshow('video', frame)

    if cv2.waitKey(1) == 27:
        cap.release()
        cv2.destroyAllWindows()
        break
```



Step 2 訓練取樣集。

將 h0 資料夾中的圖片取出、標籤化後送進人臉特徵演算法中計算特徵值，並將結果存檔後以供後續使用。

1-8-2 使用 LBPH 訓練。(1-8-2-train.py)

```
import cv2
import numpy as np

images = []
labels = []

for index in range(100):
    filename = 'images/h0/{:03d}.pgm'.format(index)
    print('read ' + filename)
    img = cv2.imread(filename, cv2.COLOR_BGR2GRAY)
    images.append(img)
    labels.append(0)    # 第一張人臉的標籤為 0

print('training…')
model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(images), np.asarray(labels))
model.save('faces.data')
print('training done')
```

Step 3 辨識。

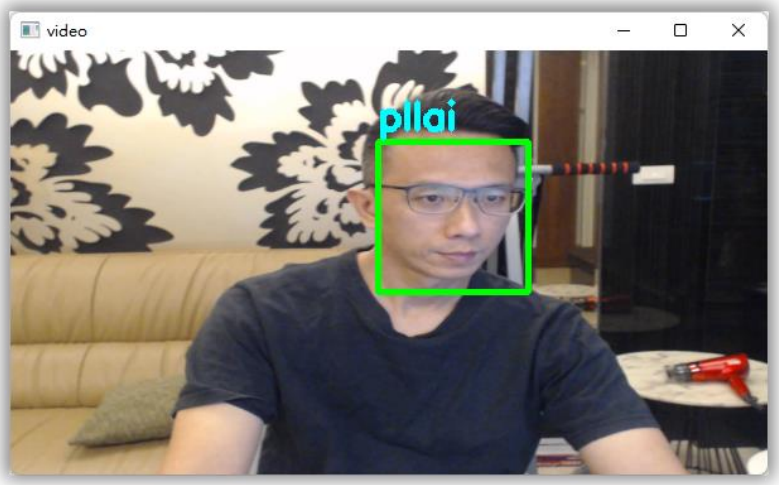
這一階段是拿訓練好的結果，來驗證辨識效果是否準確，所以我們開啟攝影機，將攝影機拍到的人臉來跟訓練結果比對，如果發現有『 認識 』的人臉，就在畫面上顯示這個人臉的名字。

1-8-3 即時人臉辨識。(1-8-3-recognition.py)

```
import cv2

model = cv2.face.LBPHFaceRecognizer_create()
model.read('faces.data')
print('load training data done')

# 載入聯集分類器（需與取樣時的分類器一致），並開啟攝影機
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
cv2.namedWindow('video', cv2.WINDOW_NORMAL)

# 可識別化名稱
names = ['第一位的名字']

# 讀取攝影機資料，，並轉換成灰階影像進行辨識
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (600, 400))
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 3)

    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x, y), (x + w, y + h),
                              (0, 255, 0), 3)
        face_img = gray[y: y + h, x: x + w]
        face_img = cv2.resize(face_img, (400, 400))

        val = model.predict(face_img)
        print('label:{}, conf:{:.1f}'.format(val[0], val[1]))

        if val[1] < 50:
            cv2.putText(frame, names[val[0]], (x, y - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)

    cv2.imshow('video', frame)
```

```
    if cv2.waitKey(1) == 27:
        cap.release()
        cv2.destroyAllWindows()
        break
```

輸出結果：



1-5. 整合專案

整合專案 1：多個特定人臉辨識

很多場合都需要特定人臉辨識，例如：人臉門禁系統、身分認證、機器人識別系統。此節介紹如何進行多個人臉的模型訓練與辨識。（專案資料夾名稱：project_multiFace）

Step 1 取樣階段。(1-9-1-capture_multiface.py)

預先新增好 h0（第一位的人臉資料夾）、h1（第二位的人臉資料夾）…，依此類推。取樣時只需更改 **1-8-1** 的 saveImage() 的資料夾名稱，手動完成多位的人臉取樣。如下所示：

```
def saveImage(face_image, index):
    filename = 'images/h0/{:03d}.pgm'.format(index)
    cv2.imwrite(filename, face_image)
    print(filename)
```

Step 2 訓練取樣集。(1-9-2-train_multiface.py)

現在資料夾 imges 裡有多個人臉樣本的資料夾，所以需要修改 **1-8-2**，同時訓練多張人臉。

1-9-2 使用 LBPH 訓練多張人臉。

```
import cv2
import numpy as np

images = []
labels = []
path = './images'

for file in os.listdir(path):
    fullname = os.path.join(path, file)
    if os.path.isdir(fullname):
        print('{} <DIR>'.format(file))
        for index in range(100):
            filename = 'images/{}/{:03d}.pgm'.format(file, index)
            print('read ' + filename)
            img = cv2.imread(filename, cv2.COLOR_BGR2GRAY)
            images.append(img)
            labels.append(int(file))
    else:
```

```
        print('{}'.format(file))

print('training…')
model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(images), np.asarray(labels))
model.save('faces.data')
print('training done')
```

Step 3 辦識。(1-9-3-recognition_multiface.py)

只需修改 **1-8-3** 的可識別化名稱，加入多個人的名稱到 name (list 容器)，即可完成多人辨識。

```
# 可識別化名稱
names = ['第一位的名字', '第二位的名字']
```

整合專案 2：物聯網應用，將目標影像回傳 — 以 LINE 為例

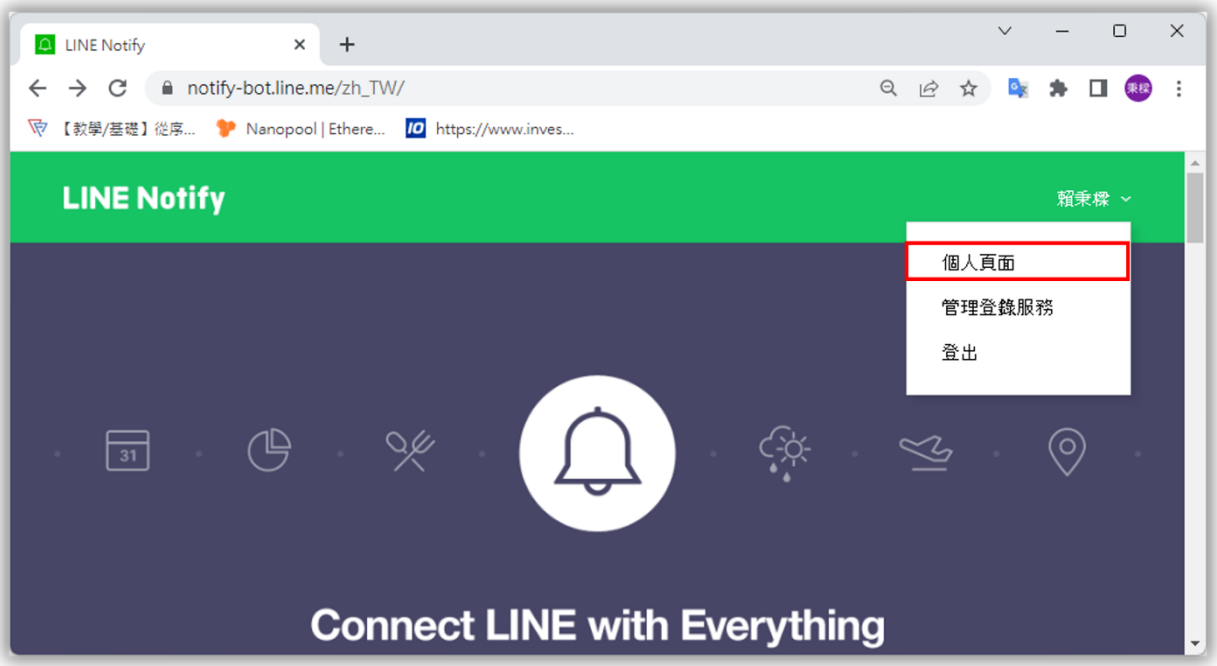
物聯網的中心是當 『 物 』 透過網路之後，可以實現出哪些該物的應用。目前已完成即時人臉辨識，再加入 LINE 的通知，將目標物辨識出來後，也能即時的通知。(專案資料夾名稱：project_multiFace-LINE)

申請 LINE Notify 的權杖

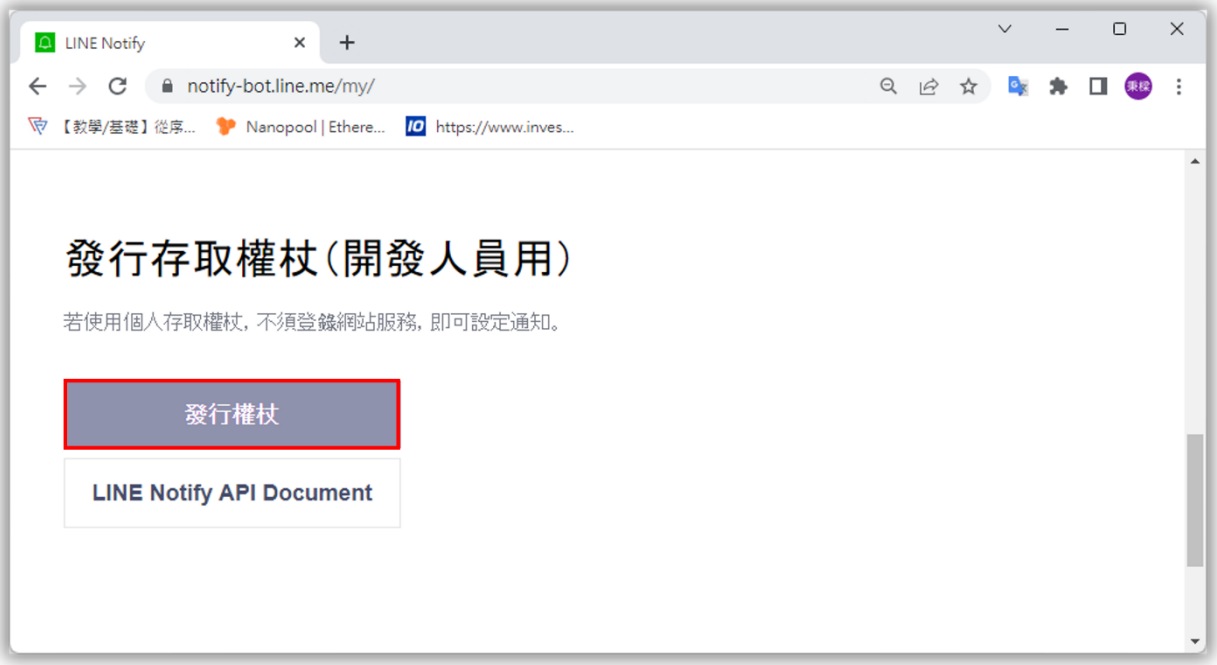
官網：https://notify-bot.line.me/zh_TW/

進入 LINE Notify 的網站後，輸入帳密登入後，並申請一個權杖 (token)。

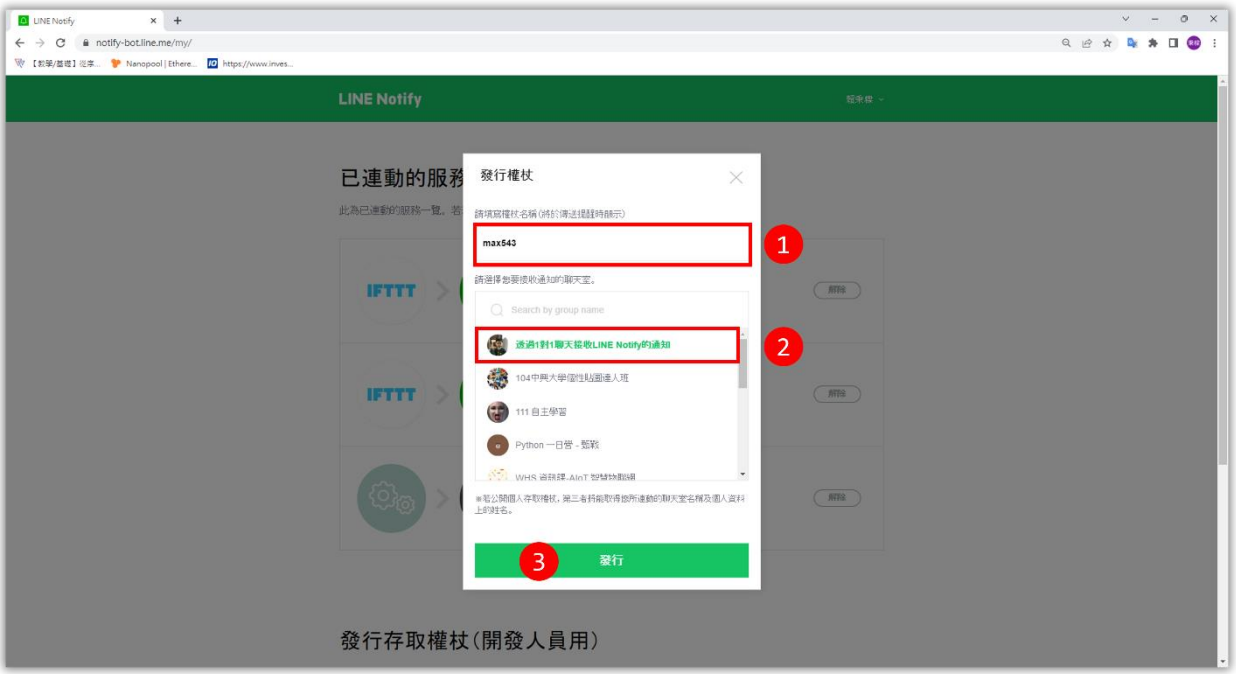
Step 1 點選『 個人頁面 』。



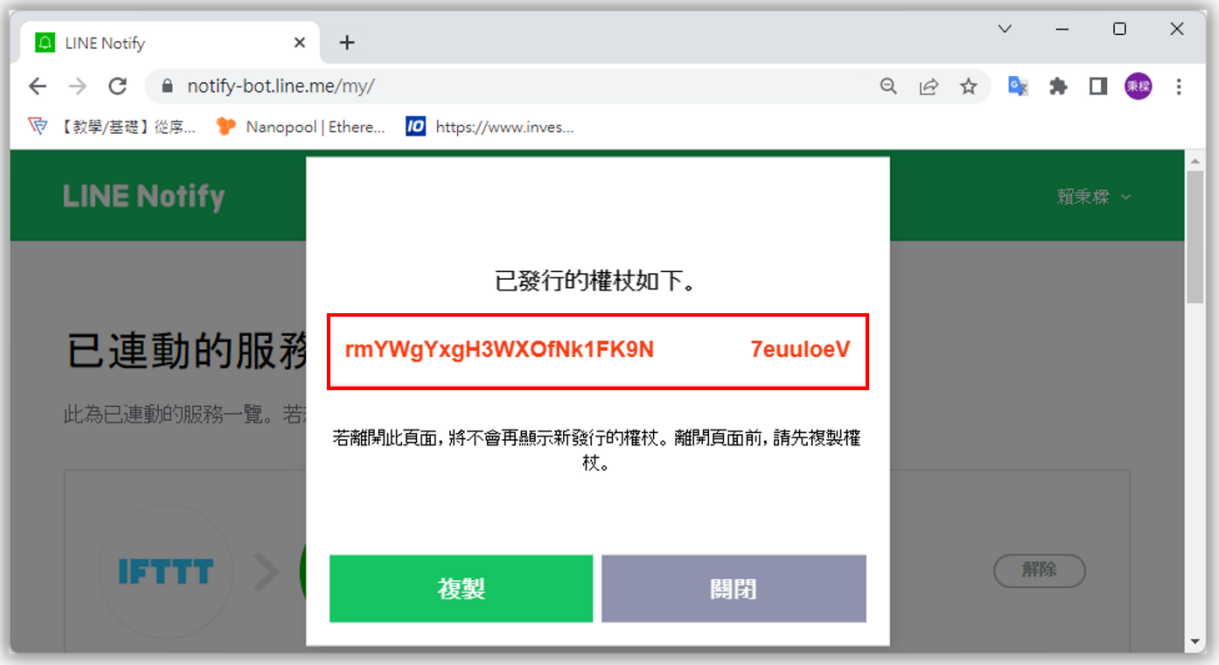
Step 2 點選『 發行權杖 』。



Step 3 輸入一個自訂的『 權杖名稱 』，並選擇『 透過 1 對 1 聊天接收... 』，測試發送給自己，完成後點擊『 發行 』。



Step 4 複製你的權杖。(記得複製保存，目前關閉後無法再次查到)



Step 5 設定 LINE 的傳送程式。(line.py)

將你的權杖修改成你的權杖碼。

line 使用 LINE Notify 的推播。

```
import requests

# LINE Setup
token = '你的權杖'

def send_msg():
    url = 'https://notify-api.line.me/api/notify'
    headers = {'Authorization':'Bearer '+ token}
    data = {'message':'注意: 有地球人出現!'}
    image = 'img/image.jpg'
    file = {'imageFile': open(image, 'rb')}
    r = requests.post(url, headers = headers, params = data,
                      files = file)
```

- line.py 是一個自訂的模組，使用時需要 import line。
- 模組內含一個 send_msg() 函式，呼叫時不需要傳入任何參數。

1-10-3 即時人臉辨識。(1-10-3-recognition.py)

```
import cv2, line

model = cv2.face.LBPHFaceRecognizer_create()
model.read('faces.data')
print('load training data done')
# 載入聯集分類器（需與取樣時的分類器一致），並開啟攝影機
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
cv2.namedWindow('video', cv2.WINDOW_NORMAL)
# 可識別化名稱
names = ['第一位的名字', '第二位的名字']
# 讀取攝影機資料，並轉換成灰階影像進行辨識
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, (600, 400))
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 3)

    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x, y), (x + w, y + h),
                              (0, 255, 0), 3)
        face_img = gray[y: y + h, x: x + w]
        face_img = cv2.resize(face_img, (400, 400))
        val = model.predict(face_img)
        print('label:{}, conf:{:.1f}'.format(val[0], val[1]))
        if val[1] < 50:
            cv2.putText(frame, names[val[0]], (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)
                    cv2.imwrite('img/image.jpg', frame)
            if n % 100 == 0:
                line.send_msg()
                print('Send to Line!')
            n += 1

    cv2.imshow('video', frame)
    if cv2.waitKey(1) == 27:
        cv2.destroyAllWindows()
        break
```

1-6. MediaPipe 機器學習框架

Google MediaPipe

MediaPipe 是 Google 公司在 2019 年發表的開放原始碼專案，此專案針對即時串流媒體和電腦視覺 (Computer Vision)，提供開放原始碼且跨平台的機器學習解決方案，這個解決方案就是機器學習管線 (ML Pipeline)。

基本上，Google MediaPipe 是一種圖表基礎系統 (Diagram Based System)，可以用來建構多模式影片、聲音和感測器等應用的機器學習管線，我們可以使用圖形方式來組織模組元件，例如：TensorFlow 或 TensorFlow Lite 推論模型和多媒體處理函數等，來建構出一個擁有感知功能的機器學習管線，能夠即時從媒體中辨識出人臉、手勢和姿勢等感知功能，其官方網址如下所示：

官網 https://mediapipe.dev/

安裝 MediaPipe

新增一個 Python 3.9 的虛擬環境 mediapipe，務必先安裝 OpenCV 後，繼續安裝 Google MediaPipe：

離開目前的虛擬環境，例如：opencv。

```
(opencvenv_39) >>> deactivate
```

MediaPipe 也會到 OpenCV 相同的模組，所以從已建立的虛擬環境 opencv，複製出一個虛擬環境 mediapipe，再進行後續安裝：

```
>>> conda create -n mediapipeenv_39 --clone opencvenv_39
```

在樹莓派 4 安裝 MediaPipe 的指令：

```
(mediapipeenv_39) >>> pip install mediapipe
```

MediaPipe 人臉偵測

MediaPipe 偵測臉部六個關鍵點

MediaPipe 人臉偵測 (Face Detection) 是使用 Blazeface 模型的一種超快速的人臉偵測，Blazeface 模型是 Google 開發的一種快速和輕量級的人臉辨識模型，可以在圖片中辨識出多張人臉和標示臉部 6 個關鍵點 (Key Points)，這是使用 Single Shot Detector 架構和客製化編碼器所建立的人臉辨識模型。

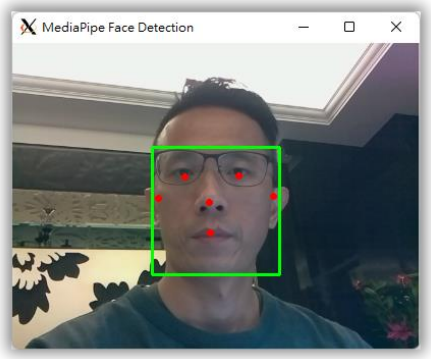
MediaPipe 人臉偵測所辨識出的臉部可以回傳臉部範圍的方框座標，再加上左眼、右眼、鼻尖、嘴巴、左耳和右耳共 6 個關鍵點座標。

1-11 MediaPipe 人臉偵測與臉部六個關鍵點。

```
import cv2
import mediapipe as mp

ESC = 27

mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)
face_detection =
mp_face_detection.FaceDetection(min_detection_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame.flags.writeable = False
    results = face_detection.process(frame)
    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    if results.detections:
        for detection in results.detections:
            mp_drawing.draw_detection(frame, detection)
```

```
    cv2.imshow('MediaPipe Face Detection', frame)

    if cv2.waitKey(1) == ESC:
        break

cap.release()
cv2.destroyAllWindows()
```

輸出結果：



MediaPipe 臉部網格

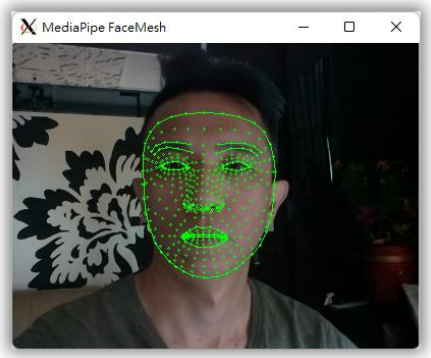
MediaPipe 臉部網格 (MediaPipe Face Mesh) 是使用 Blazeface 模型為基礎，可以預測出 468 個關鍵點，和使用網格來繪出 3D 臉部模型。

1-12 MediaPipe 臉部網格。

```
import cv2
import mediapipe as mp

ESC = 27

mp_drawing = mp.solutions.drawing_utils
mp_face_mesh = mp.solutions.face_mesh

drawing_spec = mp_drawing.DrawingSpec(thickness = 1, circle_radius = 1)

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)
face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence = 0.5,
                                  min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame.flags.writeable = False
    results = face_mesh.process(frame)
    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    if results.multi_face_landmarks:
        for face_landmarks in results.multi_face_landmarks:
            mp_drawing.draw_landmarks(image = frame,
                                      landmark_list = face_landmarks,
                                      connections = mp_face_mesh.FACE_CONNECTIONS,
                                      landmark_drawing_spec = drawing_spec,
                                      connection_drawing_spec = drawing_spec)
    cv2.imshow('MediaPipe FaceMesh', frame)
```

```
    if cv2.waitKey(1) == ESC:
        break

cap.release()
cv2.destroyAllWindows()
```

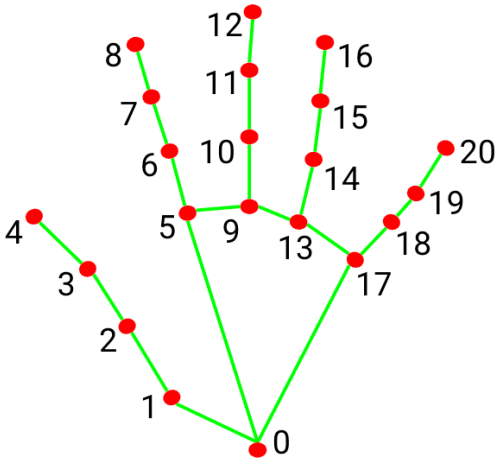
輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 臉部網格進行人臉辨識和繪出 3D 臉部模型，程式執行的結果可以看到綠色線標示出的 3D 臉部網格。

MediaPipe 多手勢追蹤

MediaPipe 手勢 (MediaPipe Hands) 是使用手掌偵測模型 (Palm Detection Model) 進行多手勢追蹤，首先偵測出手掌和拳頭，然後使用手部地標模型 (Hand Landmark Model) 偵測出手部的 21 個關鍵點，如下圖所示：



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

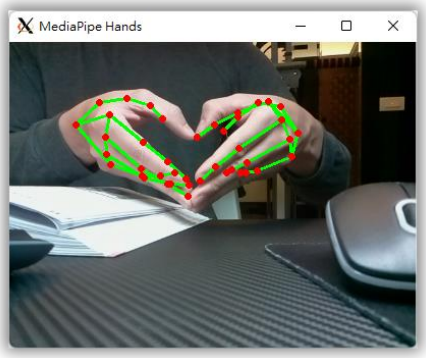
官網 https://google.github.io/mediapipe/solutions/hands.html

1-13 MediaPipe 多手勢追蹤。

```
import cv2
import mediapipe as mp

ESC = 27

mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

hands = mp_hands.Hands(min_detection_confidence = 0.5,
                                        min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame.flags.writeable = False
```

```
    results = hands.process(frame)
    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, hand_landmarks,
                                      mp_hands.HAND_CONNECTIONS)

    cv2.imshow('MediaPipe Hands', frame)

    if cv2.waitKey(1) == ESC:
        break

cap.release()
cv2.destroyAllWindows()
```

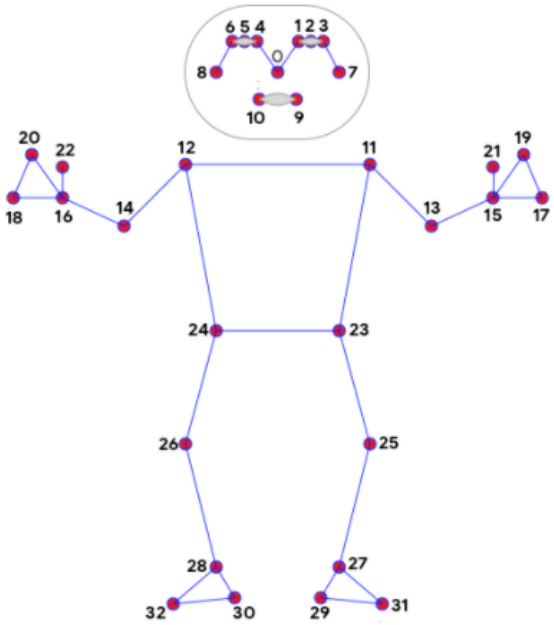
輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 多手勢追蹤進行手勢辨識，程式執行的結果可以看到綠色線標示出手部地標的 21 個關鍵點 (紅色點)。

MediaPipe 人體姿態估計

MediaPipe 姿勢 (MediaPipe Pose) 是使用 BlazePose 偵測模型來進行人體姿態估計 (Human Pose Estimation)，首先偵測出人體後，使用人體地標模型 (Pose Landmark Model, BlazePose GHUM 3D) 偵測出人體的 33 個關鍵點，如下圖所示：



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

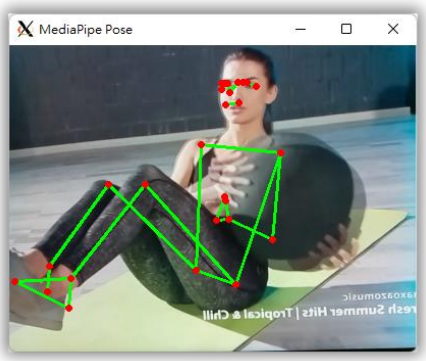
官網 https://google.github.io/mediapipe/solutions/pose.html

1-14 MediaPipe 人體姿態估計。

```
import cv2
import mediapipe as mp

ESC = 27

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

cap = cv2.VideoCapture(0)
ratio = cap.get(cv2.CAP_PROP_FRAME_WIDTH) /
cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
WIDTH = 400
HEIGHT = int(WIDTH / ratio)

pose = mp_pose.Pose(min_detection_confidence = 0.5,
                                        min_tracking_confidence = 0.5)

while cap.isOpened():
    ret, frame = cap.read()
```

```
    frame = cv2.resize(frame, (WIDTH, HEIGHT))
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame.flags.writeable = False
    results = pose.process(frame)
    frame.flags.writeable = True
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    mp_drawing.draw_landmarks(frame, results.pose_landmarks,
                                            mp_pose.POSE_CONNECTIONS)
    cv2.imshow('MediaPipe Pose', frame)

    if cv2.waitKey(1) == ESC:
        break

cap.release()
cv2.destroyAllWindows()
```

輸出結果：



- Python 程式在使用 OpenCV 讀取影像後，使用 MediaPipe 人體姿態估計進行姿勢的辨識，程式執行的結果可以看到綠色線標示出人體地標的 33 個關鍵點（紅色）。

練習 1

1. 修改程式 1-14.py 修改成辨識跳躍或雙手舉高的姿態後，繪製框框或在影格中標註文字。